Statistical Techniques in Robotics (16-831, S21) Lecture #05 (Monday, February 15)

Online Linear Classification (Perceptron, Winnow)

Lecturer: Kris Kitani *Scribes: Yen-Chi Cheng, Hao-Ming Fu*

1 Review

In the last lectures, we relax the realizability assumption and covered the two algorithms: Weighted Majority Algorithm (WMA) and Randomized Weighted Majority Algorithm (RWMA).

The Weighted Majority Algorithm (WMA) learns a weight for each expert and use their weighted opinion to make predictions. Also, the weights are updated according to the performance of their corresponding experts. To be more precise, in each round, the weights of the experts who fail to provide correct advice are lowered by a factor $(1-\eta)$. The algorithm is as follows:

Algorithm 1 Weighted Majority Algorithm (WMA)

$\boldsymbol{w}^{(1)} \leftarrow \{w_n^{(1)} = 1\}_{n=1}^N$ ▷ Weight initialization
$\eta \leq \frac{1}{2}$ ▷ Penalty rate initialization
for $t = 1, \cdots, T$ **do**
  RECEIVE $(\boldsymbol{x}^{(t)} \in \{-1, 1\}^N)$ ▷ Receive expert predictions
  $\hat{y}^{(t)} = \text{sign}\Big(\sum_{n=1}^N x_n^{(t)} \cdot w_n^{(t)}\Big) \in \{-1, 1\}$ ▷ Make learner prediction
  RECEIVE $(y^{(t)} \in \{-1, 1\})$ ▷ Receive actual answer
  $w_n^{(t+1)} \leftarrow w_n^{(t)}\big(1 - \eta \cdot \mathbf{1}[y^{(t)} \neq x_n^{(t)}]\big)$ ▷ Weight update
end for

The upper bound of mistakes made by WMA is as follows:

$$M^{(t)} \leq \frac{2\log N}{\eta} + 2m_n^{(t)}(1+\eta) \quad \forall n$$

Also, we derived the upper bound of the regret of WMA:

$$R(h_n) = M^{(T)} - m_n^{(T)} \leq m_n^{(T)} + 2\eta m_n^{(T)} + \frac{2\log N}{\eta} \tag{1}$$

By considering possible choices of η, we concluded that the average regret of WMA doesn't converge to zero over time and, thus, WMA is not a no-regret algorithm.

Then, we introduced Randomized Weighted Majority Algorithm (RWMA). The difference between RWMA and WMA is that RWMA samples a single expert to follow rather than using the weighted opinions of the experts. Specifically, the weights of the experts serve as the probability distribution of the sampling process. The algorithm is as follows.

Algorithm 2 Randomized Weighted Majority Algorithm (RWMA)

$\mathbf{w}^{(1)} \leftarrow \{w_n^{(1)} = 1\}_{n=1}^{N}$ ▷ Weight initialization
$\eta \leq \frac{1}{2}$ ▷ Penalty rate initialization
for $t = 1, \cdots, T$ **do**
  RECEIVE $(\mathbf{x}^{(t)} \in \{-1, 1\}^N)$ ▷ Receive experts predictions
  $I \sim \text{MULTINOMIAL}(\mathbf{w}^{(t)}/\Phi^{(t)})$, where $\Phi^{(t)} = \sum_{n=1}^{N} w_n^{(t)}$
  $\hat{y}^{(t)} = h_i(\mathbf{x}^{(t)})$ ▷ Make learner prediction via sampling
  RECEIVE $(y^{(t)} \in \{-1, 1\})$ ▷ Receive actual answer
  $w_n^{(t+1)} \leftarrow w_n^{(t)}\big(1 - \eta \cdot \mathbf{1}[y^{(t)} \neq h_n(\mathbf{x})^{(t)}]\big)$ ▷ Weight update
end for

We calculated the upper bound of expected mistakes and obtained:

$$\mathbb{E}[M^{(T)}] \leq (1 + \eta) m_n^{(T)} + \frac{\log N}{\eta}$$

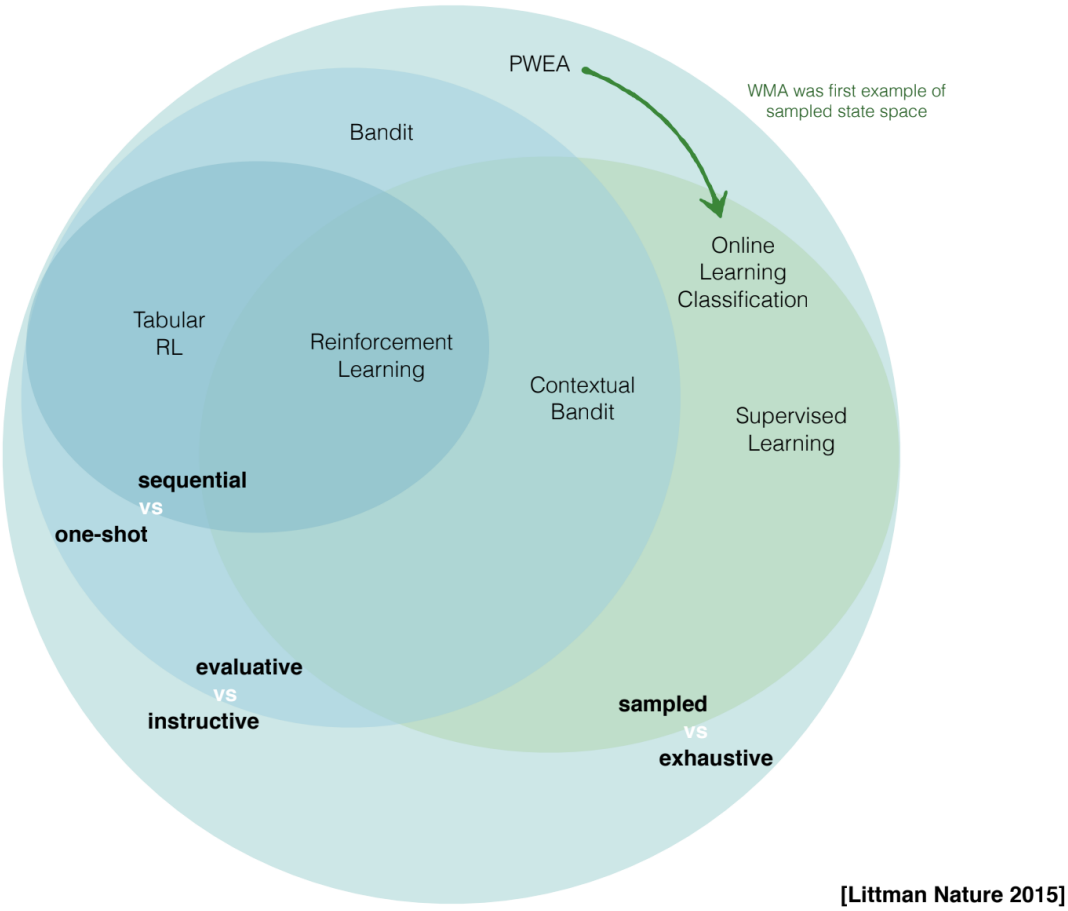
Following this, we derived the upper bound of expected regret of RWMA:

$$\mathbb{E}[R] = \mathbb{E}[M^{(T)}] - m_n^{(T)} \leq \eta m_n^{(T)} + \frac{\log N}{\eta} \tag{2}$$

With this, we noticed that if we set η to $\frac{1}{\sqrt{T}}$, the average regret of RWMA converges to zero over time. Thus, RWMA is a no-regret algorithm.

2 Online Linear Classification

Linear classification algorithms are usually allowed to make use of all available data altogether. On the other hand, in an online learning scenario, a learning algorithm has to consider the training samples one by one. Here we will include two online linear classification algorithms, the Perceptron algorithm and the Winnow algorithm. They are similar in many ways, while using different approaches to update the model parameters and assumes different assumptions. Online Linear Classification falls into the category of Online Learning Classification, and the comparison between PWEA could be seen below:



[Littman Nature 2015]

3 Perceptron Algorithm

3.1 Overview

The Perceptron Algorithm is an algorithm proposed by Rosenblatt [2]. It can be used to learn a online linear classifier, and the algorithm is shown at Algorithm 3. Intuitively, we could understand this algorithm as finding a "hyperplane" that separates the instance $\mathbf{x}^{(t)}$ in a high dimensional space R^N. The weight $w^{(t)}$ is the normal vector to that hyperplane, and will be updated with Eq. 6 at Algorithm 3 once it makes a mistake. Note that we assume the data is linearly separable.

Algorithm 3 Perceptron Algorithm

1: $\boldsymbol{w}^{(1)} \leftarrow \mathbf{0}$ ▷ Weight initialization
2: **for** $t = 1, \cdots, T$ **do**
3: RECEIVE $(\boldsymbol{x}^{(t)} \in \mathbb{R}^N)$ ▷ Receive expert predictions
4: $\hat{y}^{(t)} = \text{sign}\Big(\langle \boldsymbol{w}^{(t)}, \boldsymbol{x}^{(t)} \rangle\Big)$ ▷ Make learner prediction
5: RECEIVE $(y^{(t)} \in \{-1, 1\})$ ▷ Receive actual answer
6: $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{w}^{(t-1)} + y^{(t)} \cdot \boldsymbol{x}^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$ ▷ Weight update
7: **end for**

The update value $y^{(t)} \cdot x^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$ could either be positive or negative, as both $y^{(t)}$ or $x^{(t)}$ could be negative.

3.2 Properties of the Perceptron Algorithm

We discuss some of the properties regarding the Perceptron Algorithm.

1. *Is it fast?* **Yes.** The prediction is a dot product and a sign function, and the update rule only involves a dot product and a sum when it makes a mistake.
2. *Is it a Large Margin classifier?* **No**. It only looks at the sign of the dot product, so there is no notion of margin in the Perceptron Algorithm.
3. *Does it work on non-separable data?* **No.** "Non-separable" means that we cannot find a linear decision boundary that perfectly separates the data. Therefore, it cannot work on non-separable data. However, when the data is linearly separable, the Perceptron Algorithm will make a finite number of mistakes [1].

3.3 Mistake Bound

We derive the mistake bound for the Perceptron Algorithm. Recall the 5-step strategy we used on previous algorithms: 1) define "**potential**" function, 2) **upper bound** the potential function, 3) **lower bound** the potential function, 4) combine bounds, 5) Algebra/approximation to get performance bound. Note that we do not derive the regret bound since we assume the linear separability.

Theorem 1. *(Mistake bound of the Perceptron Algorithm) Let M be the total mistakes made by the Perceptron Algorithm, $R = \max_x \|\boldsymbol{x}^{(t)}\|$ is the norm of the observations, and $\gamma = \min_x y_t \langle \boldsymbol{w}^* , \boldsymbol{x}^{(t)} \rangle$ is the margin of separability. Then M is bounded by:*

$$M \leq \frac{R^2}{\gamma^2} \tag{3}$$

Proof. **Upper bound of the potential function.** We start by defining the potential function as:

$$\Phi^{(t)} = \|\boldsymbol{w}^{(t)}\|^2 = \sum_{n=1} \left(w_n^{(t)}\right)^2 \tag{4}$$

with some derivation, we have

$$\Phi^{(t)} = \|\boldsymbol{w}^{(t-1)} + y^t \boldsymbol{x}^{(t)}\|^2 = \sum_{n=1} \left(w_n^{(t)}\right)^2 , \tag{5}$$

$$= \|\boldsymbol{w}^{(t-1)}\|^2 + \|\boldsymbol{x}\|^2 + 2y^{(t)} \langle \boldsymbol{w}^{(t-1)} , \boldsymbol{x}^t \rangle . \tag{6}$$

since $2y^{(t)}\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^t \rangle$ must be negative because the learner makes a mistake at time t. Hence,

$$\begin{aligned}
\Phi^{(t)} &= \|\boldsymbol{w}^{(t-1)}\|^2 + \|\boldsymbol{x}\|^2 + 2y^{(t)}\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^t \rangle, && (7)\\
&\leq \|\boldsymbol{w}^{(t-1)}\|^2 + \|\boldsymbol{x}\|^2, && (8)\\
&\leq \|\boldsymbol{w}^{(t-1)}\|^2 + R^2. && (9)
\end{aligned}$$

Equation 9 is obtained by replacing $\|\boldsymbol{x}\|$ with $R = \max_x \|\boldsymbol{x}^{(t)}\|$.
And the upper bound could be obtained using induction. Starting from the base case,

$$\begin{aligned}
\Phi^{(1)} &\leq \|\boldsymbol{w}^{(0)}\|^2 + R^2, && \text{(1 mistakes)}\\
\Phi^{(2)} &\leq \|\boldsymbol{w}^{(0)}\|^2 + 2R^2, && \text{(2 mistakes)}\\
&\vdots\\
\Phi^{(T)} &\leq \|\boldsymbol{w}^{(0)}\|^2 + M^{(T)}R^2, && \text{(M mistakes)}\\
\Phi^{(T)} &\leq M^{(T)}R^2 && (10)
\end{aligned}$$

So we obtain the **upper bound** for the potential function.

Lower bound of the potential function. The lower bound is a little tricky. We adopt an strategy to first define an intermediate potential $\langle \boldsymbol{w}^*, \boldsymbol{w}^{(t)} \rangle$, where $\boldsymbol{w}^*$ is the perfect classifier and is a unit vector, and then upper bound and lower bound this dot product to derive the lower bound for $\Phi^{(t)}$. We start from the upper bound for this dot product. Observe that,

$$\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} = \operatorname*{argmax}_{\boldsymbol{w}'} \langle \boldsymbol{w}', \boldsymbol{w} \rangle \tag{11}$$

since the dot product is largest when two vectors are oriented in the same direction. Note that $\boldsymbol{w}^*$ is a unit vector, the dot product is maximized when:

$$\langle \boldsymbol{w}^*, \boldsymbol{w}^{(T)} \rangle \leq \langle \frac{\boldsymbol{w}^{(T)}}{\|\boldsymbol{w}^{(T)}\|}, \boldsymbol{w}^{(T)} \rangle = \frac{\|\boldsymbol{w}^{(T)}\|^2}{\|\boldsymbol{w}^{(T)}\|} = \|\boldsymbol{w}^{(T)}\|. \tag{12}$$

Therefore the upper bound is $\langle \boldsymbol{w}^*, \boldsymbol{w}^{(T)} \rangle \leq \|\boldsymbol{w}^{(T)}\|$. We proceed the proof by deriving the lower bound for $\langle \boldsymbol{w}^*, \boldsymbol{w}^{(t)} \rangle$. By applying the update rule to $\boldsymbol{w}^{(t)}$,

$$\langle \boldsymbol{w}^*, \boldsymbol{w}^{(t)} \rangle = \langle \boldsymbol{w}^*, \boldsymbol{w}^{(t-1)} \rangle + y^{(t)} \langle \boldsymbol{w}^*, \boldsymbol{x}^{(t)} \rangle. \tag{13}$$

note that $y^{(t)}\langle \boldsymbol{w}^*, \boldsymbol{x}^{(t)} \rangle$ is always positive because $\boldsymbol{w}^*$ is the perfect classifier. Plug in $\gamma = \min_x y_t \langle \boldsymbol{w}^*, \boldsymbol{x}^{(t)} \rangle$,

$$\begin{aligned}
\langle \boldsymbol{w}^*, \boldsymbol{w}^{(t)} \rangle &= \langle \boldsymbol{w}^*, \boldsymbol{w}^{(t-1)} \rangle + y^{(t)} \langle \boldsymbol{w}^*, \boldsymbol{x}^{(t)} \rangle,\\
\langle \boldsymbol{w}^*, \boldsymbol{w}^{(t)} \rangle &\geq \langle \boldsymbol{w}^*, \boldsymbol{w}^{(t-1)} \rangle + \gamma, && (14)
\end{aligned}$$

Now we could derive the lower bound by applying induction. Start from the base case,

$$\begin{aligned}
\langle \boldsymbol{w}^*, \boldsymbol{w}^{(1)} \rangle &\geq \langle \boldsymbol{w}^*, \boldsymbol{w}^{(0)} \rangle + \gamma,\\
\langle \boldsymbol{w}^*, \boldsymbol{w}^{(2)} \rangle &\geq \langle \boldsymbol{w}^*, \boldsymbol{w}^{(0)} \rangle + 2\gamma,\\
&\vdots\\
\langle \boldsymbol{w}^*, \boldsymbol{w}^{(T)} \rangle &\geq \langle \boldsymbol{w}^*, \boldsymbol{w}^{(0)} \rangle + M^{(T)}\gamma. && (15)
\end{aligned}$$

since the initial value $\langle \boldsymbol{w}^* , \boldsymbol{w}^{(0)} \rangle = 0$, we have the lower bound for the dot product:

$$\langle \boldsymbol{w}^* , \boldsymbol{w}^{(T)} \rangle \geq M^{(T)} \gamma \,. \tag{16}$$

Finally, we can combine the upper and lower bound for $\langle \boldsymbol{w}^* , \boldsymbol{w}^{(T)} \rangle$ (Equation 12 and 16) and derive the lower bound for the potential function. Recall that the upper bound $\langle \boldsymbol{w}^* , \boldsymbol{w}^{(T)} \rangle \leq \|\boldsymbol{w}^{(T)}\|$ is the square root of the potential $\Phi^{(T)}$, we have:

$$\begin{aligned} \langle \boldsymbol{w}^* , \boldsymbol{w}^{(T)} \rangle \geq M^{(T)} \gamma \;\; , \langle \boldsymbol{w}^* , \boldsymbol{w}^{(T)} \rangle \leq \|\boldsymbol{w}^{(T)}\| \,, \\ \|\boldsymbol{w}^{(T)}\| \geq M^{(T)} \gamma \,, \\ \|\boldsymbol{w}^{(T)}\|^2 \geq (M^{(T)} \gamma)^2 \,. \end{aligned} \tag{17}$$

thus the lower bound to the original potential function is $\Phi^{(T)} \geq (M^{(T)}\gamma)^2$.

Combining bounds. Finally, we combine the upper and lower bound of the potential function:

$$\begin{aligned} \Phi^{(T)} \leq M^{(T)} R^2 \;\; , \Phi^{(T)} \geq (M^{(T)} \gamma)^2 \,, \\ (M^{(T)} \gamma)^2 \leq M^{(T)} R^2 \,, \\ M^{(T)} \leq \frac{R^2}{\gamma^2} \,. \end{aligned} \tag{18}$$

□

Interpreting the mistake bound. The margin γ represents the degree to which the data is separable. When γ is large, M becomes smaller since it is easier for the learner to classify the data. On the other hand, the norm R represents a "hyperball" that contains all observations. When R is large, it becomes harder for the learner to handle the instance. Therefore, M becomes smaller as well.

4 Winnow Algorithm

4.1 Overview

The Goal of the Winnow Algorithm is to learn a linear classifier from labeled data. It's applied to problems whose inputs are binary features and outputs (labels) are binary predictions/labels. It can be used for both online and offline settings.

The Winnow Algorithm assumes that the problem can be represented as a disjunctive boolean function. That is, only a certain subset of the input features are relevant to the classification task. It means that the output is actually the logical OR of the relevant features.

4.2 Winnow Algorithm

A single sample x can be denoted as $\{x_1, x_2, ..., x_N\}$, where x_i is the value of a single boolean feature and there are N input features. The learning process of the Winnow Algorithm is as follows.

Algorithm 4 Winnow Algorithm

1: $\boldsymbol{w}^{(1)} = \{1, 1, ..., 1\}$ ▷ weight initialization
2: **for** $t = 1, \cdots, T$ **do**
3: RECEIVE $(\boldsymbol{x}^{(t)} \in \{0, 1\}^N)$
4: $\hat{y}^{(t)} = \mathbf{1}[\langle w^{(t)}, x^{(t)} \rangle > N]$ ▷ make prediction
5: RECEIVE $(y^{(t)} \in \{0, 1\})$
6: $w_i^{(t+1)} = w_i^{(t)}(1+\beta)^{(y^{(t)} - \hat{y}^{(t)}) \cdot x_i^{(t)}}$ ▷ update the weights
7: **end for**

where $y^{(t)}$ is the label for $\boldsymbol{x}^{(t)}$.

The Winnow Algorithm is similar to the Perceptron Algorithm, with the following differences.

Algorithm 5 Perceptron Algorithm

1: $\boldsymbol{w}^{(1)} \leftarrow 0$ ▷ Winnow initializes the weights to 1s
2: **for** $t = 1, \cdots, T$ **do**
3: RECEIVE $(\boldsymbol{x}^{(t)} \in \mathbb{R}^N)$ ▷ Winnow observations are binary
4: $\hat{y}^{(t)} = \text{sign}\Big(\langle \boldsymbol{w}^{(t)}, \boldsymbol{x}^{(t)} \rangle\Big)$ ▷ Winnow uses N
5: RECEIVE $(y^{(t)} \in \{1, -1\})$ ▷ Winnow uses binary output
6: $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} + y^{(t)} \cdot x^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$ ▷ Winnow uses exponential updates
7: **end for**

4.3 Explanation

From a high level point of view, the classifier learns a weight for each attribute and uses the sum of the weights of triggered (by a sample) attributes to make a prediction. Specifically, if the sum of the weights is greater than a predefined threshold, which is N in this case, the classifier predicts 1, otherwise 0.

During the learning process, whenever a prediction is incorrect, update the weights (by multiplication) of the attributes triggered by the sample toward the direction of the true label. For instance, if a sample triggers attribute x_1, x_3, x_5 and the classifier predicts 0 but the true label is 1, the weights of the attributes, x_1, x_3, x_5 are multiplied by $(1 + \beta)$. On the other hand, if the classifier predicts 1 but the true label is 0, the weights of the attributes, x_1, x_3, x_5 are multiplied by $1/(1+\beta)$.

The number of mistakes of the Winnow Algorithm, M, has an upper bound:

$$M < 2 + 3k(\log_2 N + 1) \tag{19}$$

where k is the number of relevant features.

5 Conclusion

The goal of the online learning classification is for the autonomous system, where experts will work along side and give the 'correct' classification. It aims to facilitate the autonomous system to learn a good classifier over time. We covered two online learning classification algorithm: the Perceptron Algorithm and the Winnow Algorithm. The Perceptron Algorithm adopted additive approach for its update rule. It is fast in both aspect of prediction and update, but have no notion of regret. Assuming linear separability, the number of mistakes will converge. On the other hand, the Winnow Algorithm uses multiplicative rule for updating the weight, and is adopted to learn the logical OR function of k relevant attributes.

References

[1] A. B. Novikoff. On convergence proofs for perceptrons. Technical report, Stanford Research Inst Menlo Park CA, 1963. 4

[2] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. 3